Release: 2.00

Virus Scan Adapter Test and Certification Program

Readme for Virus Scan Adapter Test and Certification Program (VSATEST)

History

Version	Status	Date
1.0	Release	22.07.2005
1.1	Release	26.08.2005
2.0	Release	19.12.2012

Dietmar-Hopp-Allee 1
D-69190 Walldorf

Copyrights and Trademarks

Most recent version available:

http://www.sap.com/corporate-en/legal/copyright/index.epx

Partners for NW-VSI should contact icc@sap.com and/or copyrights@sap.com

Contents

Overview

SAP Virus Scan Adapter Test and Certification Program (VSATEST) is a tool for testing and evaluation of SAP Virus Scan Adapters. It is an essential tool in developing Virus Scan Adapter and SAP highly recommends on using it prior coming to our premises for certification check.

Package Content

The VSATEST package has the following layout:

opt/

This folder has some precompiled VSATEST binaries. Provided executables are non-Unicode and multi-thread enabled. Binaries for other supported by SAP platforms are available on request.

etc/

This folder has a sample XML configuration and test case files (*vsa-cert.xml and vsa-smoke.xml*).

vsa-cert.xml: can be used to check if the VSA under development would eventually pass the certification check done by SAP Certification Center when the time comes.

vsa-smoke.xml: this is a quick smoke test of the VSA adapter. It checks if clean and infected files are handled properly and it is also a good starting point to write your own test cases.

The syntax of these files is explained in the next chapters.

NOTE: There are some hard coded paths in the XML files. If you plan to run VSATEST from within different layout you may have to adjust those paths as necessary. The paths can be absolute or relative. The slashes in the path are automatically converted according to the run-time platform so you may use in the XML file whichever you like.

log/

This folder is empty by default but if you use the provided XML configurations, a trace debug output files will be placed here.

out/

This folder is empty by default but if you use the provided XML configurations the output files will be placed here.

data/

This is the folder where VSATEST expects to find all need test data files. There are two sub-folders here – clean (contains only clean files) and infected (contains files infected with EICAR test virus)

shakespeare
This is actually plain text file that contains part of Shakespeares Sonnet 18 and can be considered clean (virus free). Naturally it can be replaced by any other file as long as it is detected as clean by the virus scan engine(s) used by the VSA.

eicar.txt
This is a file specially developed by the Europen Institute of Anti-Virus Research for the purpose of testing anti-virus products. To read more about this test-virus as well as to download a sample of eicar.com visit http://www.eicar.org/ and from “General Information” menu click on “The AntiVirus testfile eicar.com”. Additionally you may want to check SAP OSS note 666568.

Installation

The VSA test suite comes as part of VSA SDK package. Please read VSA SDK readme document for information on how to install it. After installation make sure the extracted VSATEST folder has the same content as described in the previous chapter.

After unpacking the SDK content find int the vsatest root folder go.sh (Unix) and go.cmd (Windows) batch scripts that can be used right away to evaluate how VSATEST utilty works.

Configuration

VSATEST configuration is done through mix of environment variables, command line options and XML files.

NOTE: In the examples bellow <VSATEST> is to be replaced with the actual binary name (e.g. vastest on Unix/Linux and vastest.exe on Windows).

Environment Variables

Setting environment variables is usually system dependent. Here are some examples:

On Microsoft Windows platforms (standart DOS-shell):

> ***set <VARIABLE>=<VALUE>***

On Unix and Linux platforms (csh, tcsh and compatible):

> ***set <VARIABLE>=<VALUE>***

On Unix and Linux platforms (bash and compatible):

> ***export <VARIABLE>=<VALUE>***

NOTE: To understand how to add, change and remove environment variables please consult your favourite shell documentation.

The following environment variables are recognized an used if set:

***VSA_LIB* –** sets the path to the VSA library that VSATEST will try to load.

Command Line Options

List of command line options is always available by running:

> **> <*VSATEST*> *help***

Here is a description of the currently available command line options:

-cfg <file name>
Sets name of configuration XML file to be used for VSA and server settings. If no command is provided, the file will be used as input configuration. If 'get_config' command is used then VSA configuration will be saved to this file.

> Default: *vastest.xml*
>
> Examples:
>
> To save current VSA configuration:
>
> **> <*VSATEST*> get_config -cfg config.xml**
>
> To specify input configuration:
>
> **> <*VSATEST*> *-cfg test.xml***

-i <file name>
Sets name of a test case file. A test case file has XML syntax and specifies the test procedures to be executed by VSATEST. Information on how to define your own test case files is discussed in the following chapters.

> Default: *vastest.xml*
>
> Examples:
>
> **> <*VSATEST*> -i test-case-1.xml**
>
> **> <*VSATEST*> -cfg config.xml -i TC1.xml**

-o <file name>
Sets the name of the output XML file where results from the run will be saved. The output file has XML syntax and contains summary of the function calls executed by VSATEST. Can be feeded to CSS, XLT, etc. files for additional parsing as needed.

> Default: *vsaout.xml*
>
> Examples:
>
> **> <*VSATEST*> -o vsatest-out.xml**

-f <file name>
Sets the name of the file where trace output will be placed. Through trace file and trace level (see bellow) the VSA adapter work can be debugged on a low level. Output is a plain text suitable to be explored with regular text editor.

Default: *vsatrace.trc*

Examples:

> <*VSATEST*> -f /var/log/vsatrace.log

> <*VSATEST*> -f c:\logs\vsatrace.log

-l <0|1|2|3>
Sets the trace level. This is the severity of the messages logged as follows:

0: log only error messages
1: log error and warning messages
2: log error and warning messages and calls to virus scan engine
3: log everything available including memory allocations and deallocation

Default: *0*

Examples:

> <*VSATEST*> -f /var/log/vsatrace.log –l 3

-c <code page number>
Sets the code page that will be used.

Default: *1100*

Examples:

> <*VSATEST*> -c 1111

-V <path to VSA lib>
Sets the path to the VSA library to be loaded. If both this option and environment variable 'VSA_LIB' is provided then the first one is ued.

Default: *none*

Examples:

> <*VSATEST*> -V c:\vsa\vsa.dll

> <*VSATEST*> -V /usr/local/lib/vsa.so

-p <profile name>
Sets the profile name for the current VSA configuration.

Default: *VSA_CONFIG*

Examples:

> <*VSATEST*> get_config -cfg config.xml -p VSA_CONFIG_1

-debug
Set debug brakepoint. This is usefull to attach debug to external VSA adapter.

Default: *off*

Examples:

> <*VSATEST*> -debug

-abort
Default flag for VSA callback messages.

Default: *off*

Examples:

> <*VSATEST*> -abort

-info
Show extra information about the loaded VSA.

Default: *off*

Examples:

> <*VSATEST*> -info

will printout something like this:

```
Virus Scan Adapter (VSA)
        Version         : 1.0
        Adaptername : VSA dummy for tests
        Vendorinfo     : SAP AG, Walldorf (Germany)
        Parameters     : 2
        Parameter 1   | SCANBESTEFFORT          | BOOL   |  0
        Parameter 2   | SCANALLFILES            | BOOL   |  1
Virus Scan Engine
        Version       : 1.0
        Versiontext   : Dummy Adapter without AV protection
        Engine date   : Sat Aug 21 01:00:00 2004
        Known viruses : 1
        Loaded drivers: 1
                Driver 1   | Dummy driver only for EICAR-STRING
```

-vsa <node name>
Set the XML node name where all VSA settings are stored. Through this option you can tell vastest to use different settings.

Default: *vsa*

Examples:

> <*VSATEST*> -vsa VSA2

-t <node name>
Test case XML node name. This option is useful to tell VSATEST to run only specific test case. Otherwise all tests will be runned.

Default: *run all test cases*

Examples:

> <*VSATEST*> -t TEST-CASE-01

-T <1-999>
Maximum amount of threads to start. This option is valid only if “mt” is set to true in a function XML node. Use this feature with caution, as several handles will be allocated which had to be released afterwards. For example: If you use "mt" in VsaScan definition then you have to use it also in VsaReleaseScan definition.

Default: *1*

Examples:

> <*VSATEST*> -T 3

-m <1-999>
Minimum threads the process should keep. This option is only useful for the Virus Scan Server and therefore has no relevance for VSATEST.

Default: *1*

Examples:

> <*VSATEST*> -m 5

-r <number>
Loop counter. Use this option to set how many times VSATEST shall loop on running test case(s). This option is useful for e.g. stress tests.

Default: *1*

Examples:

> <*VSATEST*> -cfg vastest.xml -r 10

-q
Quiet mode. Use this option to suppress any extra output and if you are only interested in the overall result.

Default: *off*

Examples:

> <*VSATEST*> -cfg vastest.xml -q

-v
Verbose mode. Use this option to enable printing of detailed otput information.

Default: *off*

Examples:

> <*VSATEST*> -cfg vastest.xml -v

NOTE: If quiet mode is enabled then verbose mode is automatically disabled.

-D
Test data directory. This option is only valid for the command *testdata*.

Default: *data*

Examples:

> <*VSATEST*> testdata -D myTest

XML configuration profiles

VSATEST can be completely configured through profiles kept in XML files. There can be one or many configuration profiles in a single XML file that can be selected via '-vsa' command line option as needed. An example configuration profile looks like this:

```
<vsa>
 <parameter name="VSA_LIB" type="CHAR"></parameter>
 <parameter name="VSA_Profile" type="CHAR">DEFAULT</parameter>
 <parameter name="Test_Profile" type="CHAR"></parameter>
 <parameter name="TraceMaxlines" type="CHAR">10000</parameter>
 <parameter name="TraceMaxhold" type="CHAR">86400</parameter>
 <parameter name="TraceBinaryLevel" type="CHAR">0</parameter>
 <parameter name="TestCaseFile" type="CHAR">vsa-cert.xml</parameter>
 <parameter name="OutputFile" type="CHAR">out/vsa-cert.xml</parameter>
 <parameter name="TraceLevel" type="CHAR">0</parameter>
 <parameter name="TraceFile" type="CHAR">log/vsa-cert.log</parameter>
 <parameter name="MinThreads" type="CHAR">1</parameter>
 <parameter name="MaxThreads" type="CHAR">1</parameter>
</vsa>
```

Parameters are self explanatory and directly mapped to the corresponding command line options.

Test case definition

Here is an example of a case that tests the ability of VSA to scan clean files.

```
 <test name="VSA-CERT-01" description="Scanning clean file">
  <function name="VsaStartup" rc="VSA_OK"/>
  <function name="VsaGetConfig" rc="VSA_OK">
   <parameter name="pVsaConfig">VSA-SCAN-CFG</parameter>
  </function>
  <function name="VsaInit" rc="VSA_OK">
   <parameter name="pVsaInit">VSA-SCAN-INIT</parameter>
  </function>
  <function name="VsaScan" mt="false" rc="VSA_OK">
   <parameter name="pVsaInit">VSA-SCAN-INIT</parameter>
   <parameter name="pVsaScanParam">
          <parameter name="struct_size">28</parameter>
          <parameter name="tScanCode">VSA_SP_FILE</parameter>
          <parameter name="tActionCode">VSA_AP_SCAN</parameter>
          <parameter name="pszObjectName">data/shakespeare.exe</parameter>
          <parameter name="pbByte"></parameter>
          <parameter name="lLength"></parameter>
          <parameter name="pVsaScanInfo">VSA-SCAN-INFO</parameter>
   </parameter>
  </function>
  <function name="VsaReleaseScan" mt="false" rc="VSA_OK">
   <parameter name="pVsaScanInfo">VSA-SCAN-INFO</parameter>
  </function>
  <function name="VsaEnd" mt="false" rc="VSA_OK">
   <parameter name="pVsaConfig">VSA-SCAN-CFG</parameter>
   <parameter name="pVsaInit">VSA-SCAN-INIT</parameter>
  </function>
  <function name="VsaCleanup" rc="VSA_OK"/>
 </test>
```

As one can see the XML parameters maps directly to the C interface thus making it really easy to define different types of test cases. You may define as much test cases as you like. The order of execution is the same as in the XML files.

Evaluation of VSATEST output

Console output

While running, VSATEST generates certain amount of console output. Messages are printed to standart error output (stderr). For each VSA function calls there are two actions performed. One is to evaluate the return code and one is to certify the returned (output) parameter(s). For each action an overall result is printed, either as “**PASS**”, “**WARN**” or “**FAIL**”.

There are three types of output generated:

- *Quiet output*

For each action printed is only the final result and nothing else.

- *Normal output*

For each action printed is the final result. For return code evaluation printed is the expected return codes and the actual one. For the certification action printed are the parameters evaluated and any possible error or warning messages.

- *Verbose output*

For each action printed is the final result. For return code action reported is also the time spent in the function call (in microseconds) for performance evaluation. For the certification action information about the parameter values is printed as well as error and warning messages.

After VSATEST completes running the test cases a summary is printed:

- *Number of loops* – how many times test case(s) were repeated.
- *Number of tests* – how many unique test cases were executed.
- *RC Errors* – how many return code errors were found.
- *RC Warnings* – how many return code warning were found.
- *CERT Errors* – how many certification errors were found.
- *CERT Warnings* – how many certification warning were found.

NOTE: For input parameters no actual certification is done, meaning possible errors in input parameters are always reported as warnings.

To assume that the VSA would eventually pass the certification check at SAP run of VSATEST with the default vsa-cert.xml configuration file must not report any RC Errors or CERT Errors. In the best case no warning shall be printed as well but that is not mandatory.

XML output

And output XML file is generated if so required. It has very much the same information as the verbose console output but represented in XML syntax. This output file might then be translated to HTML file for example for own purposes.

Trace output

Trace output is plain text file reporting in details the internal work of the VSATEST program. The amount of trace information logged is controlled via command line parameter ("-l") or from the input XML configurationfile. The output of this file shall only be useful for VSA developers.

Known problems

There are currently no known problems. If you find any please report them to the responsible persons bellow. Do not forget to mention platform, operating system and exact steps to reproduce.

Contacts

Please report any questions, suggestiong and problems about VSATEST suite to the authors directly:

Hristo Grigorov <hristo.grigorov@sap.com>
Markus Strehle <markus.strehle@sap.com>